arXiv:1509.00556v1 [cs.SI] 2 Sep 2015

Efficient Detection of Communities with Significant Overlaps in Networks: Partial Community Merger Algorithm

Elvis H. W. Xu and P. M. Hui
Department of Physics, The Chinese University of Hong Kong, Shatin, New Territories, Hong Kong SAR, China
(Dated: April 25, 2016)

The identification of communities in large-scale networks is a challenging task to existing searching schemes when the communities overlap significantly among their members, as often the case in large-scale social networks. The strong overlaps render many algorithms invalid. We propose a detection scheme based on properly merging the partial communities revealed by the ego networks of each vertex. The general principle, merger criteria, and post-processing procedures are discussed. This partial community merger algorithm (PCMA) is tested on two modern benchmark models. It shows a linear time complexity and it performs accurately and efficiently when compared with two widely used algorithms. PCMA is then applied to a huge social network and millions of communities are identified. A detected community can be visualized with all its members as well as the number of different communities that each member belongs to. The multiple memberships of a vertex, in turn, illustrates the significant overlaps between communities that calls for the need of a novel and efficient algorithm such as PCMA.

PACS numbers: 89.75.Fb, 89.65.Ef, 89.75.Hc, 89.20.Ff

I. INTRODUCTION

Community structure is commonly found in networked systems in nature and society [1]. While it is almost common sense to realize the existence of communities, extracting such mesoscopic structures efficiently and accurately remains a challenging task and yet it is crucial to the understanding of the functionality of these systems. Although the definition of community remains ambiguous and a commonly accepted definition is lacking, many detection algorithms have been developed in the past decade [2] with most of them based on the concept that there should be more edges within the community than edges connecting to the outside [1, 3, 4]. This viewpoint on what a community is about implies that communities are disjoint, and it is behind the design of non-overlapping community detection algorithms. However, it was soon found by empirical studies that it is common for communities to overlap, i.e. each vertex may have multiple memberships [5] and thus it may be shared by communities. Several approaches have been proposed for detecting overlapping communities, including clique percolation [5], link partitioning [6–8], local expansion and optimization [9–11], and label propagation [12–14]. These methods perform well when overlapping vertices constitute a small portion of the network, but most of them fail to detect communities when the overlaps are significant [11, 13, 15]. The reason is that most of these algorithms are still based on the assumption that a community should have more internal than external edges and thus the communities are only slightly overlapping, which is invalid when the vertices have multiple memberships. For example, if all members of a community have two or more memberships, it is highly likely that the community has more edges connecting to other communities than internal edges. A new concept of community is therefore needed for cases of significant overlaps. We adopt an intuitive idea that each member of a community should be connected to a certain fraction of the other members. The idea is similar to *k-core*, but it is fraction-based and we call it *f-core*. Unlike *k-core*, an *f-core* allows its members to have multiple *f-cores* and thus it is suitable for describing cases that a member could belong to many communities. Yet, an *f-core* is not necessarily a single connected component and could sometimes be two or more separate clusters. To make it a useful concept in the context of communities with significant overlaps, a further constraint that a community has members who are densely connected to each other and thus a relatively high value of clustering coefficient will prove effective.

Structurally, communities with significant overlaps are hidden under dense and messy edges, unlike the cases of disjoint and slightly overlapping communities. Identifying such communities is highly non-trivial from a global or top-down viewpoint. This motivated us to approach the problem from a local or bottom-up viewpoint. Starting locally from a vertex, it is easier to identify which groups a vertex belongs to in the subnetwork consisting of the vertex itself and its neighbors, i.e. the ego network of a vertex, as illustrated in Fig. 1 using data from an online social network. The local view gives a natural sample of vertices and edges that allows us to visualize the hidden community structure clearly though partially. It is partial because the ego network only reveals part of each of the different communities that a vertex belongs to. The idea is then to replace a very difficult task from global approaches by many easier tasks of finding community structures from a local approach and construct a proper way to aggregate the results. This inspired the present work. Here, we propose a novel and efficient approach based on local views of the vertices for detecting communities with significant overlaps that works in linear time. Two steps are involved: detecting communities locally for each vertex and merging similar ones to recover the complete communities. Our method has many advantages. As in other algorithms based on a local approach [10], the method does not require an input on a pre-set number of communities to search for and it has a linear time complexity for sparse networks. Most importantly,

the assumption of disjoint or slightly overlapping communities is abandoned and the method is designed to handle the possibility of multiple memberships of a vertex and detect significantly overlapping communities. Our method also allows vertices to be homeless, i.e., they do not belong to any communities. This feature is very important in dealing with real-world networks, and yet very few algorithms considered this possibility [11]. Community detection algorithms must be able to distinguish real communities from pseudo communities [16, 17]. We are well aware of the issue and our method sifts out real communities by applying proper thresholds. All these advantages make the method uniquely capable of detecting communities with significant overlaps efficiently in large-scale real networks with hundreds of millions of vertices.

The plan of the paper is as follows. In Sec. II, we introduce the details of the algorithm, including similarity measure, merger of similar communities, thresholds, and applicability. In Sec. III, the method is tested against two benchmarks and its performance in accuracy and efficiency is compared with two other recently proposed algorithms. In Sec. IV, we apply the method to a large empirical data set and show that significantly overlapping communities are common in social networks. Results are summarized in Sec. V.

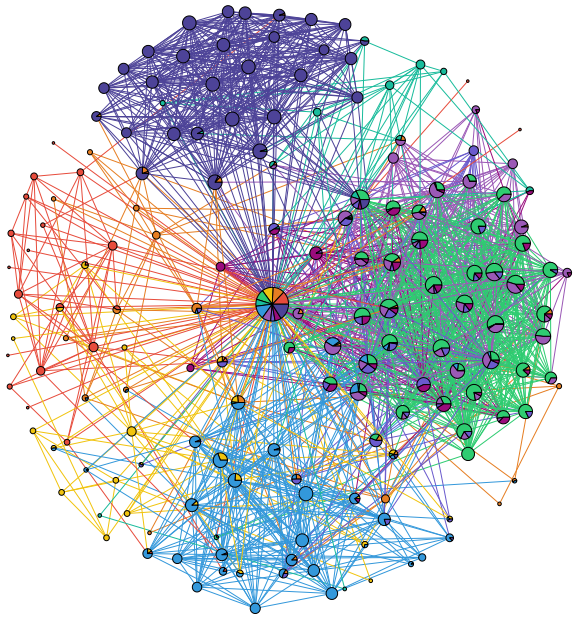

FIG. 1: (Color online) An ego network of a vertex provides the local information and reveals several partial communities. The network was constructed from data collected from Sina Weibo, an online social network akin to the hybrid of Facebook and Twitter. The partial communities are found by an existing algorithm as described in Appendix A. PCMA is an efficient and accurate algorithm for detecting complete communities in a huge network by properly merging partial communities revealed by the ego networks of all the vertices.

II. PARTIAL COMMUNITY MERGER ALGORITHM

A. General Principle

We aim to detect communities in a network in which vertices could span from being homeless to belonging to multiple communities. This renders many top-down algorithms invalid. We first give a physical picture of our algorithm. Consider a community in which every member is connected to a certain fraction of the other members. At the local level of the members, they only known their own neighbors and have no knowledge of the complete community. They are given the task of compiling a roster of the community and identify who the core members are. To complete the task, each member shares its local information consisting of a name list including itself and all its neighbors. A complete roster can in principle be derived by merging these individual name lists skillfully. Those who appear frequently on the lists are the core members, while those with less occurrence are on the periphery of the community. This merger process is the core idea of our method of detecting communities.

Practically, we start with exploring the ego network of a vertex, i.e. the subnetwork consisting of the vertex itself and its neighbors, and identifying the communities hidden in it. This is illustrated in Fig. 1 for a vertex (the central one) in an online social network. This local view lets us see the communities clearly. Since a vertex may not know all the members in each of its communities, the information on the communities found based on the ego network of a vertex is incomplete. We refer to them as the *partial communities* from the viewpoint of the vertex. This process can be carried out for every vertex. Although each member only helps reveal part of the whole picture, the idea is that aggregating local information should reveal the *complete communities*, i.e., every community with all its members. With the partial communities revealed by different vertices, we need to determine which ones are actually different parts of the same community. Merger of these partial communities in a proper way gives a complete community. It is technically difficult task as vertices may be misclassified into partial communities that they actually do not belong to. The merger process leads to much noise in the merged communities. A cleaning process or post-processing scheme is then invoked to eliminate the misclassified vertices and sift out the real and complete communities.

Our method thus consists of three steps:

1. Find the partial communities in the ego network of each of the vertices.

2. Merge partial communities that are parts of the same community to reconstruct complete communities.

3. Clean the merged communities to sift out real communities.

For easy reference, we call the method Partial Community Merger Algorithm or PCMA in short. It is a general approach. It can be implemented in different ways. For Step 1, many existing algorithms are available and we use the one proposed

by Ball *et al.* [18], with details given in Appendix A. The new elements are Step 2 and Step 3. Below, we introduce our implementation of Steps 2 and 3 in detail.

B. Merger

The merger process aims to determine whether two partial communities are part of the same complete community. This is a classic clustering problem. We thus extend an idea from agglomerative hierarchical clustering to the present purpose: Start with a set of partial communities, merge the most similar pair of communities into one and repeat the merger until the similarity is below a threshold.

Care must be taken in choosing a suitable similarity measure between communities, each is represented by a set of vertices. The Jaccard index is a common similarity measure. It is defined to be the size of the intersection divided by the size of the union of two sets. A drawback of the index is that the members are assumed equal. A merged community, however, contains core members, peripheral members, and even misclassified ones. They should be treated differently. Here we propose a novel similarity measure that incorporates the different importance of members.

For a Vertex i in Community C, let $S_{i,C}$ be a score that represents its importance in C. Without loss of generality, we define $S_{i,C} = 0$ if $i \notin C$. Let l_C be the number of partial communities that have merged to form Community C. Before the merger, the partial communities identified by Step 1 all have $l = 1$ and all members carry an initial score of 1. When two communities A and B merge into one, e.g. $C = A \cup B$, the quantities $S_{i,C}$ and l_C are given by

$$\begin{aligned} S_{i,C} &= S_{i,A} + S_{i,B} \\ l_C &= l_A + l_B \end{aligned} \tag{1}$$

Physically, $S_{i,C}$ traces the number of occurrences of Vertex i in the l_C partial communities that have merged to form Community C. Vertices with a high value of S/l are regarded as core members, and those with a small S, say less than 3, are very likely vertices that are misclassified.

Consider two communities A and B. We define an asymmetric measure $f(A, B)$ to take into account the different importance of members as:

$$f(A,B) = \sum_i \frac{S_{i,B}}{l_B} \cdot \frac{S_{i,A}}{w_A} \tag{2}$$

where $w_A = \sum_i S_{i,A}$. The term $S_{i,B}/l_B$ represents a normalized importance of Vertex i in B and $S_{i,A}/w_A$ is a weighting factor of Vertex i in A. The product $S_{i,B} \cdot S_{i,A}$ ensures that f will not be affected much by the misclassified vertices, i.e. those with small values of S. A large value of $f(A, B)$ indicates that the core members of A are also core members of B, but *not* vice versa as $f(A, B) \neq f(B, A)$ in general. This measure has the following properties:

$$f(A, B \cup C) = \frac{l_B}{l_B + l_C} f(A,B) + \frac{l_C}{l_B + l_C} f(A,C) \tag{3}$$

$$f(B \cup C, A) = \frac{w_B}{w_B + w_C} f(B,A) + \frac{w_C}{w_B + w_C} f(C,A) \ , \tag{4}$$

which can be readily shown. Let $\{A\}$ ($\{B\}$) denote the set of partial communities that form the Community A (B). It follows from Eqs. (3) and (4) that

$$\begin{aligned} f(A,B) &= f(\cup_{x\in\{A\}} x, \cup_{y\in\{B\}} y) \\ &= \sum\nolimits_{\substack{x\in\{A\} \\ y\in\{B\}}} \frac{w_x}{w_A l_B} f(x,y) \ . \end{aligned} \tag{5}$$

Recall that x and y are partial communities and thus $f(x, y)$ is the portion of members of x who are also members of y, i.e.

$$f(x,y) = \frac{\mid x \cap y \mid}{\mid x \mid} \ . \tag{6}$$

Equation (5) indicates that $f(A, B)$ is actually a weighted average of the overlap portion $f(x, y)$ over all combinations of partial communities forming A and B, i.e. with $\{(x, y), x \in \{A\}, y \in \{B\}\}$.

The merger of two communities A and B is different from either A absorbing B or B absorbing A. Thus, a symmetric analogy of $f(A, B)$ is preferred for deciding a merger. To motivate the construction of such a parameter, we introduce a measure $g(C)$ of a Community C in a way similar to Eq. (5) that compares members in the partial communities forming C:

$$g(C) = \begin{cases} 1 & \text{if } l_C = 1 \\ \sum_{\substack{x,y\in\{C\} \\ x\neq y}} \frac{w_x f(x,y)}{w_C(l_C-1)} & \text{if } l_C > 1 \end{cases} \tag{7}$$

It gives the average portion of overlap between partial communities in a merged community. Its physical meaning can be seen by considering the special case that C is an ER random network $G(n, p)$ with members randomly connected with a probability p. A partial community now consists of a vertex and all its neighbors. The expected portion of overlap between two partial communities $f(x, y)$ is roughly p, giving $g(C) \approx p$. This indicates that $g(C)$ is approximately a measure of the fraction of other members that a member is connected to. A larger $g(C)$ implies denser internal edges in the community and thus members are connected tightly to each other. It can be used as an indicator on whether a merged community is a real community or just a wrongly merged set of vertices.

For the case $C = A \cup B$, $g(C)$ satisfies

$$\begin{aligned} g(C) = \frac{1}{w_C(l_C-1)} &\{w_A(l_A-1)g(A) + w_B(l_B-1)g(B) \\ &+ (w_A l_B + w_B l_A) f_s(A,B)\} \ , \end{aligned} \tag{8}$$

where

$$\begin{aligned} f_s(A,B) &= \frac{w_A l_B f(A,B) + w_B l_A f(B,A)}{w_A l_B + w_B l_A} \\ &= \frac{2\sum_i S_{i,A} \cdot S_{i,B}}{w_A l_B + w_B l_A} \ . \end{aligned} \tag{9}$$

There are three terms in Eq. (8) for $g(C)$. The first and second terms give the overlap portions within A and within B, respectively.

The third term in Eq. (8) measures the overlap between A and B. It is important to note that a *symmetric measure* $f_s(A, B)$, as defined in Eq. (9), emerges. It is a weighted average of the asymmetric measures $f(A, B)$ and $f(B, A)$ and yet itself satisfies $f_s(A, B) = f_s(B, A)$. It follows from Eq. (9) that $f_s(A, B \cup C)$ is given by a weighted average of $f_s(A, B)$ and $f_s(A, C)$, and thus

$$f_s(A, B \cup C) \leq \max\{f_s(A, B), f_s(A, C)\} \ . \qquad (10)$$

We are thus led to apply f_s as a symmetric similarity measure between two communities that accounts for the different importance of the members.

Based on the idea of agglomerative hierarchical clustering, the merger process using f_s as the similarity measure can be implemented as follows. Given a set $\mathscr{C}$ of communities to be merged, a straightforward way is to:

1. Calculate f_s for each pair of communities in $\mathscr{C}$ and maintain a priority queue of f_s in descending order.
2. Merge the pair with the largest f_s and update the priority queue.
3. Repeat 2 until the largest f_s in the priority queue falls below a threshold t_{f_s}.

The time complexity of this algorithm is $O(n^2 \log n)$, where n is the number of communities in $\mathscr{C}$. The space complexity is $O(n^2)$ as we need to maintain the priority queue of f_s. For detecting communities in large-scale networks, a more efficient algorithm is desirable. In what follows, we propose two optimizations to reduce both the time and space complexity to $O(n)$.

We define the *best merger candidate* of a community A as

$$\mathrm{bmc}(A) = \arg\max_{X \in \mathscr{C}/A} f_s(A, X) \qquad (11)$$

We argue that the algorithm above is equivalent to:

given a set of communities $\mathscr{C}$
repeat
  choose a community A from $\mathscr{C}$
  $B \leftarrow \mathrm{bmc}(A)$
  while $\mathrm{bmc}(B) \neq A$ **do**
    $A \leftarrow B$
    $B \leftarrow \mathrm{bmc}(A)$
  end while
  if $f_s(A, B) > t_{f_s}$ **then**
    merge A and B
    remove A and B, add $A \cup B$ to $\mathscr{C}$
  end if
until no communities can be merged anymore
return $\mathscr{C}$

The algorithm makes use of the property of f_s given in Eq. (10). If A and B are the best merger candidates of each other, there does not exist a community C that gives $f_s(A, C) > f_s(A, B)$, where C can be any combination of communities in $\mathscr{C}/A$. Therefore, even if $f_s(A, B)$ is not at the top of the f_s priority queue, the merger of A and B can be moved forward since other mergers higher on the priority queue that would take part will not affect the merger of A and B. An advantage is that merges are not required to proceed in order in the algorithm, and thus there is no need to maintain the f_s priority queue. The space complexity is reduced from $O(n^2)$ to $O(n)$.

The search on $\mathrm{bmc}(A)$ is formally within the set $\mathscr{C}/A$. Practically, the search area can be reduced significantly, as most of the communities in $\mathscr{C}/A$ do not even share a single member with A in sparse networks. A good approximation is to limit the search to the partial communities from the viewpoints of A's members and the merged communities containing these partial communities. As such, the time complexity of calculating $\mathrm{bmc}(A)$ does not scale with n, providing that the community size and the number of partial communities per vertex are independent of the network size. The number of iterations of finding a pair of communities to merge should also be independent of n. The repeated loop requires a time complexity of $O(n)$. Since n usually scales linearly with the network size, it can also be regarded as the network size. We thus argue that the time complexity of our optimized merger algorithm is approximately $O(n)$. This is verified numerically in Sec. III.

C. Post-processing

After merging communities, a cleaning process is needed to handle two types of noise. First, we need to identify which merged communities are real communities and which are simply merged sets of vertices by coincidence. The latter usually contain only a small number of partial communities because they are merged by accidents. The more partial communities a merged community contains, the more likely it is a real community. Thus, the parameter l of a community can be used as a measure of whether a detected community is trustful. A way to sift out real communities is to set a threshold t_l and require all real communities to have $l \geq t_l$. The threshold can be set in many ways, e.g. setting t_l based on each community's size, as a larger community usually caries a larger l. Second, we need to identify and eliminate vertices that are misclassified into partial communities in Step 1. Recall that $S_{i,C}$ is the number of occurrences of Vertex i in l_C partial communities that formed the Community C. Roughly, the probability of Vertex i being a false member should drop with $S_{i,C}$. Thus, a threshold t_S can be set to eliminate vertices with $S < t_S$. Normally $t_S = 4$ is sufficiently stringent and it should not be less than 3. There remain vertices with $S \geq t_S$ but $S/l \approx 0$. They may still not be members since they know too few other members. The ratio S/l gives an estimate on the fraction of the other members that a member is connected to. Another criterion $S/l > t_{S/l}$ becomes useful, with $t_{S/l}$ being a threshold that requires each member be connected to at least $t_{S/l} \times 100\%$ of the other members. This criterion echoes the concept of *f-core* discussed in Sec.I. The threshold $t_{S/l}$ can either be set uniformly for all communities or individually for each community based on the value of g, which reflects the average portion that a member is connected to the

other members. Since different kinds of networks may have different community structures, the choice of $t_{S/l}$ depends on the nature of communities of a specific network under study.

D. Applicability

PCMA works under two conditions: Existence of partial communities (Step 1) and adequate overlap between partial communities for mergers (Step 2). Usually, the second condition is satisfied automatically when the network under study meets the first condition. For the first condition, the existence of partial communities from the viewpoint of a vertex requires that there are sufficient number of neighbors and a high density of edges among the neighbors, i.e., a high local clustering coefficient of the vertex.

We expect a community detected by PCMA to have the following properties:

1. Two members with common neighbors are highly likely neighbors of each other. As a consequence, the community has a relatively high value of clustering coefficient.

2. The shortest distances between most pairs of members are generally short and not longer than 3. Thus, most members are connected to each other either directly or via one/two intermediate member(s).

3. Each member is connected to at least a certain fraction of the other members.

From another perspective, these properties can be taken as a broad descriptive definition of community, and are well suited for describing communities with significant overlaps. PCMA is designed to detect this kind of communities, which are important in large-scale systems with vertices typically having multiple memberships.

III. BENCHMARKING

We tested PCMA using two benchmark models to illustrate its performance and applicability. Results are compared with two fast and accurate overlapping community detection algorithms that are among the best [15]: namely OSLOM [11] and SLPA [14].

First, a simple benchmark model in the spirit of the planted l-partition model [19] is used. The network is generated as follows:

1. Generate an ER random network of n vertices with a mean degree $\langle k\rangle$ that serves as background noise.

2. Randomly sample s vertices as a community, with s satisfying the Poisson distribution with an expected value of $\langle s\rangle$. Connect each pair of members with a probability p.

3. Repeat the step to generate $n \cdot \langle c\rangle$ communities. Here, $\langle c\rangle$ is the expected number of communities that a vertex belongs to.

This model is flexible in that the total number, size, and intra-community edge density, as well as the background noise level can be directly controlled. In addition, vertices can belong to several communities or even be homeless. There is no guarantee that there are more edges within a community than edges going out. These features make many existing community detection algorithms invalid. They reflect the challenges posed by real social networks, in which a person often simultaneously belongs to many groups, on community detection. PCMA is designed to solve the problem.

Consider a network with $n = 10^5$, $\langle k\rangle = 3$, $p = 0.3$, $\langle s\rangle = 40$, and $\langle c\rangle = 2$ generated as described. The threshold $t_{f_s} = 0.1$ is chosen for the merger process. Figure 2 shows the actual community size distributions generated by the model (diamonds). The results as detected by PCMA before (circles) and after post-processing (squares) are shown for comparison. As discussed in Sec. II C, the merged communities without post-processing are noisy. The results show a peak at a small community size due to many coincidentally merged sets of vertices, which are targeted for removal in post-processing. The results also show a bump in the distribution at large community sizes. Although these are real communities, the many misclassified vertices make their sizes bigger than their actual sizes. Hence, results before post-processing could be misleading. By setting the thresholds properly, as given in the captions of Fig. 2, the distribution after post-processing is in good agreement with the actual community size distribution.

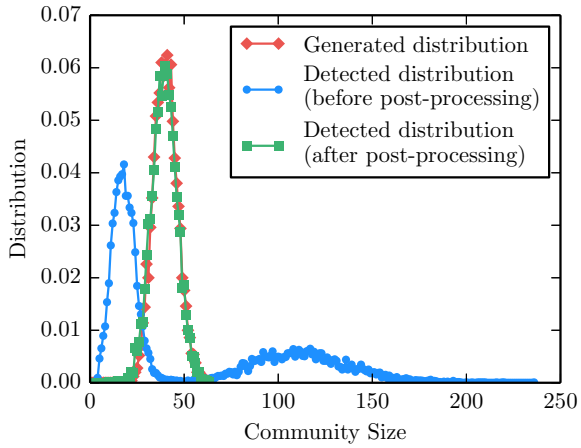

FIG. 2: (Color online) Community size distribution as degenerated by the benchmark model (diamonds) described in the text. The corresponding results as detected by our method before (circles) and after (squares) applying the post-processing scheme are shown for comparison. The thresholds in the method are set to be: $t_{f_s} = 0.1$, $t_l = 10$, $t_S = 3$, and $t_{S/l} = 0.1$.

To qualify the accuracy of PCMA, we adopt the widely used Normalized Mutual Information (NMI) [20] as extended by Lancichinetti *et al.* [10] to compare overlapping communities. Figure 3(a) compares the performance of OSLOM [11] and PCMA on synthetic networks with different intra-community edge densities. Results of SLPA are not shown because the method cannot detect homeless vertices.

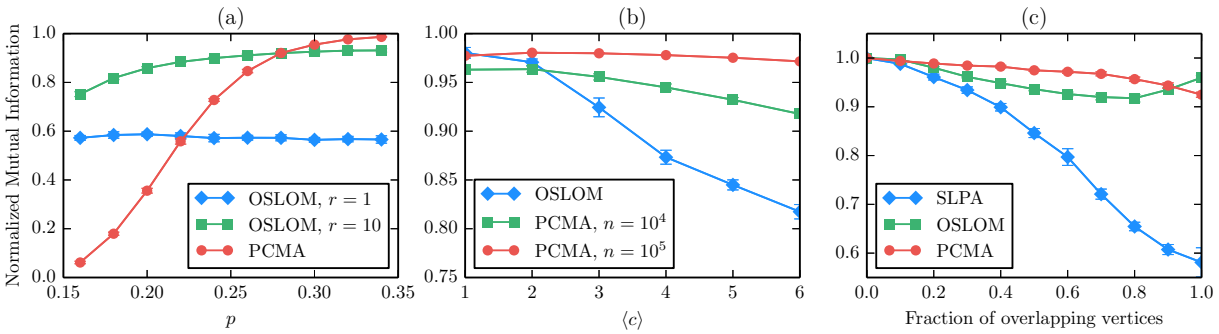

FIG. 3: (Color online) Performance comparisons of PCMA and widely used existing methods in a simple benchmark model and the LFR benchmark model. Unless stated otherwise, parameters of our simple benchmark model in (a) and (b) are: $n = 10^4$, $\langle k\rangle = 20$, $p = 0.3$, $\langle s\rangle = 40$, $\langle c\rangle = 3$. Parameters of the LFR benchmark model in (c) are: $n = 10^4$, $\langle k\rangle = 40$, $k_{max} = 100$, $\mu = 0.3$, $\tau_1 = 2$, $\tau_2 = 1$, $c_{min} = 20$, $c_{max} = 100$, each overlapping vertex has two communities. The thresholds for PCMA are the same as those given in Fig. 2. The number of iterations of OSLOM is set to $r = 10$. For SLPA, the program applies different thresholds ranging from 0.01 to 0.5 by default and we select the best result. Each data point is an average of 10 realizations. If not shown, the error bar is smaller than the size of the symbol.

The accuracy of OSLOM depends strongly on the number of iterations. We use the default value $r = 10$ suggested in Ref. [11], unless specified otherwise. For PCMA, it is sensitive to the intra-community edge density p because it affects the existence of partial communities, which is a criterion for the applicability of PCMA (see Sec. II D). In the benchmark model, the probability p promotes partial communities. PCMA works well when

$$\langle k_{nn}\rangle = [(\langle s\rangle - 1)p - 1]\, p \geq 2\ , \tag{12}$$

where $(\langle s\rangle - 1)p$ is the expected number of neighbors of a member and $\langle k_{nn}\rangle$ is the expected number of edges of a neighbor connecting to other neighbors of the member. The neighbors start to be strongly connected, i.e. partial communities emerge, when $\langle k_{nn}\rangle \geq 2$. From Fig. 3(a), PCMA performs better than OSLOM for $p \geq 0.28$, corresponding to $\langle k_{nn}\rangle \geq 2.78$ for the case of $\langle s\rangle = 40$.

Figure 3(b) shows the dependence of the performance on $\langle c\rangle$, which controls the expected number of communities that a vertex belongs to. A larger $\langle c\rangle$ corresponds to more edges connecting the communities and thus a denser and more complex network. For a system with $n = 10^4$ members, the accuracy of OSLOM falls rapidly with increasing $\langle c\rangle$. For PCMA, the accuracy remains high throughout, with a slight drop due to the finite size of the network instead of $\langle c\rangle$. This is verified by the performance of PCMA in a bigger system of $n = 10^5$ (circles in Fig. 3(b)). Recall that many existing algorithms become invalid in problems that a vertex may belong to many communities, but PCMA handles them well.

We also tested PCMA with the widely used LFR benchmark model [21]. Figure 3(c) compares the performance of PCMA with OSLOM and SLPA. In the LFR model, a vertex has a degree chosen from a distribution that follows a power-law of exponent τ_1 in a range of degrees $k_{min} \leq k \leq k_{max}$ corresponding to a mean degree $\langle k\rangle$. A tunable fraction of vertices are chosen to belong to more than one communities. They are the overlapping vertices. The remaining vertices have only one community. For a vertex of degree k, a parameter μ sets the fraction of the edges to be connected to vertices outside the community(ies) that the vertex belongs to. The remaining fraction $(1 - \mu)$ of edges are evenly divided among the communities, if the vertex is chosen to have multiple communities. As such, the community sizes also follow a power-law with an exponent τ_2 within a range of communities sizes between c_{min} and c_{min}. The combinations of parameters in the LFR model give a class of tunable structures for the resulting networks. From Fig. 3(c), all the three methods work very well when there are very few overlapping vertices. When communities overlap more, PCMA performs better than the other two methods over a wide range of the fraction of overlapping vertices, except for the last data point in Fig. 3(c) in comparison with OSLOM. In the LFR model, the degree assignment does not distinguish single-community vertices from multi-community vertices. As a result, a vertex belonging to multiple communities has fewer edges connecting to each of its communities. In PCMA, however, members are expected to be connected to at least a certain fraction of the other members before establishing their membership. This leads to the gradual drop in PCMA's NMI with increasing number of overlapping vertices, which are members according to the benchmark model but may not be identified by PCMA. We remark that it is actually not a problem of accuracy, but more about what a community should be.

We also studied the time complexity of the three methods numerically based on the LFR benchmark model [21]. Calculations were performed on a workstation with Intel Xeon E5-2609 @ 2.4GHz (4 cores / 8 threads). The programs were allowed to use all threads if they were parallelized. Figure 4 shows how the execution time scales with the network size. SLPA and PCMA are almost equally fast and OSLOM is slower by at least 500 times. In the log-log plot in Fig. 4, the slopes for SLPA, OSLOM and PCMA are 1.10, 1.09 and 1.00, respectively. It is, therefore, numerically verified that the time complexity of PCMA is $O(n)$.

In summary, the benchmark tests showed that PCMA is an efficient algorithm specifically suitable for detecting communities in networks in which the vertices may belong to multiple communities.

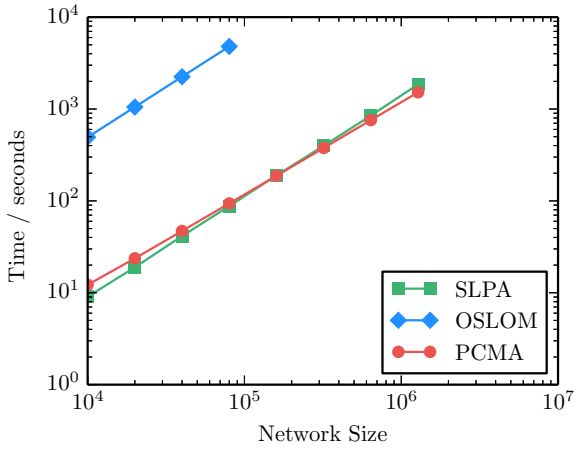

FIG. 4: (Color online) Comparison on time complexity. Tests are conducted on LFR benchmark model. The fraction of overlapping vertices is set to 50%. Other parameters are the same as those in Fig. 3. Each data point is an average over 10 realizations. Error bars are smaller than the size of the symbols.

IV. EMPIRICAL ANALYSIS

Having established the efficiency and accuracy of PCMA, we apply it to analyze data in a huge social network in China, Sina Weibo, akin to a hybrid of Twitter and Facebook. It is a directed network like Twitter for fast information spreading and it also has characteristics of Facebook for interactions with friends. We focused on the embedded undirected friendship network with only reciprocal edges and applied PCMA to extract its community structures. The network we sampled from the Internet contains about 80 million vertices and 1.0 billion reciprocal edges, with only 1.2% of the edges being connected to vertices that are not sampled. The sampled network can thus be roughly regarded as the core of the whole network. There are many vertices that have only a few neighbors and we do not expect to find partial communities by searching over these vertices. Instead, we only need to search the ego networks of vertices with a degree equal or higher than a specific value, which is taken to be 20 here. Some abnormal vertices with extremely high clustering coefficients are also omitted. It only took PCMA about 35 hours to complete the detection in such a huge network and found millions of communities on an ordinary workstation.

To illustrate the properties of the communities, Fig. 5 shows two histograms for the communities as a function of their value of g and community size, for two different thresholds $t_l = 10$ and 2. Each vertical cut gives the distribution of g for communities of the same size. The values in each cut are rescaled by mapping the highest value to unity. Usually a larger community would have a lower g, since the number of friends a person could have is limited and does not grow linearly with the community size. The results in Fig. 5(a) illustrate this relationship and show that a member typically knows $20 - 30\%$ of the other members in communities that are not too big. The results also show that PCMA is a successful algorithm in that most of the detected communities have relatively large values of g, indicating that they are real communities. Choosing a low threshold of $t_l = 2$ gives an abnormal plunge in g at small community sizes as shown in Fig. 5(b). The low threshold leads to many false communities that are merged only a few times, as discussed in Sec. II C. Raising t_l from 2 to 10 reduces the number of detected communities from 4.6 million to 0.9 million and removes most false communities. This success is accompanied by the drawback that real communities with less than 10 vertices are also removed. We also modified Eq.(9) slightly to deal with the empirical data better. Although the threshold $t_{f_s} = 0.1$ is sufficiently harsh for large partial communities, it can easily be satisfied by small ones. For example, two partial communities of size 10 only need a single common member to meet the criterion. Therefore, we suppress such unwanted mergers of small partial communities by forcing $f_s(A, B) = 0$ if $\sum_i S_{i,A} \cdot S_{i,B} / \max\{l_A, l_B\} < t_{f0}$. We used $t_{f0} = 4$ in our analysis.

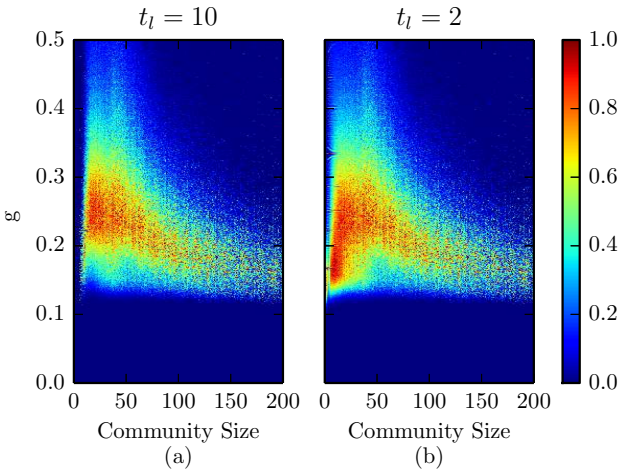

FIG. 5: (Color online) PCMA was applied to detect communities in a huge data set collected from a social network. The resulting communities have different sizes and g (see Eq. (7)). Histograms of g and community size among the detected communities are shown for two different thresholds t_I. Unless otherwise stated, parameters are the same as those given in Fig. 2.

The detected communities also confirm the multiple memberships of a vertex and thus the significant overlap in communities. Figure 6 gives a complete community that is only partially revealed in the ego community in Fig. 1. The community is found by merging $l = 49$ partial communities. After mergers, it has 287 members and most of them are misclassified. Setting thresholds $t_{S/l}$ and t_S in post-processing removes misclassified members, leaving the detected community with 58 members. It has a value of $g = 0.50$, implying that on average every member knows about half of the other members. The community actually shows a core-periphery structure, i.e., there is a group of key members knowing most members and many peripheral members knowing only the key members. The number on a vertex in Fig. 6 gives the number of communities that the vertex belongs to. Most vertices have

multiple memberships and some of them even belong to more than 10 communities. The number of edges going out of the community is 5676, much larger than the number of reciprocal edges 733×2 within the community (counted twice). This also confirms that one should not assume a community to have more internal edges than external edges when there are significant overlaps between communities.

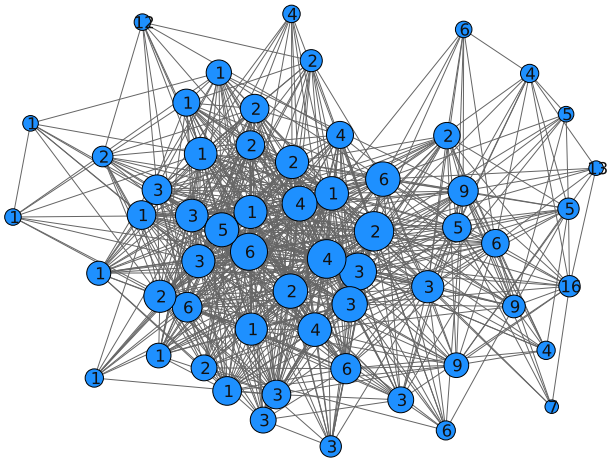

FIG. 6: A real community consisting of 58 members as revealed by applying PCMA to real social network data set. The label on a vertex gives the number of different communities that the vertex belongs to. The number of reciprocal edges within the community (counted twice) is 733×2, and the number of edges (not shown) connecting to the outside is 5676.

V. SUMMARY

We proposed and implemented a Partial Community Merger Algorithm specifically designed for detecting communities in big data sets in which a member may have multiple memberships. The structure of these communities is signified by a strong overlap in members and thus a community may have many edges connecting to the outside compared to those within the community. Such structures make many existing community searching algorithm invalid, but yet they often show up in real-world systems. Through PCMA, we provided a conceptual framework as well as a practical algorithm in dealing with these systems and supplemented the toolbox in community detection in complex network science. Details of implementing PCMA were discussed. We used two benchmark models and compared results with two widely used algorithms to establish the validity and accuracy of PCMA. The method does not need a prior knowledge of the number of communities to search for and it is capable of analyzing communities in large-scale networks in linear time. The algorithm is applied to a huge social network data set. In addition to identifying the communities, PCMA also gives who the key members are in a community and how many different communities a member belongs to. The high accuracy and linear time complexity makes PCMA a promising tool for detecting communities with significant overlaps in huge social networks, which cannot be handled by most existing algorithms.

We end with a few remarks. Although we described and implemented PCMA only for unweighted networks, the approach is flexible and it can be readily extended to treat weighted networks. Another extension is to properly tune the threshold t_{f_s} for exploring the hierarchical structure of communities. Like any other algorithm, PCMA also has its limitations. It is not designed to detect small communities and it will not work in networks that are too sparse. There is also the common problem among algorithms on distinguishing real communities from false ones. This is actually a deeper question because whether there really exists a clear boundary for distinguishing "real communities" from "false ones" is questionable. A more practically approach would be to explore methods of choosing the thresholds properly or construct a function involving g, l, and community size to make the detected communities more trustful. Finally, the source code of our implementation of PCMA is released as free software under the GNU General Public License version 2 or any later version (GPLv2+) [22].

Acknowledgements

One of us (EHWX) gratefully acknowledges the support from the Research Grants Council of the Hong Kong SAR Government through a Hong Kong PhD Fellowship.

Appendix A: Searching for Partial Communities

Before carrying out PCMA, we need to search for partial communities in the ego network of each vertex. We adopt an efficient community detection algorithm proposed by Ball *et al.* [18] for the purpose. Starting from an ego network of size n, the algorithm takes the number K of communities to be found as an input. The output is an $n \times K$ matrix, with the K numbers in a row that signifies a vertex being the belonging coefficients [23] of the vertex for the K communities. For example, we set out to find 5 communities in an ego network, labelled by $C1$ to $C5$. Then each row has 5 numbers, e.g. $0.64, 0.29, 0, 0.07, 0$ for the j-th row, denoting that the vertex j has a portion of 64% belonging to $C1$, 29% to $C2$, and 7% to $C4$. To convert the fuzzy assignment [23] of members to definite assignment, we impose a threshold that a community carries a vertex j as its member only if the belonging coefficient of the vertex j for the community is above 0.20. In the example, only communities $C1$ and $C2$ have the vertex j as a member. For the central vertex of the ego network, as the communities are its partial communities, it is treated as a default member of all the communities regardless of the threshold. Partial communities can then be derived from every ego network. We remark that although the number of partial communities in an ego network is not known in prior to the search, it can easily be estimated for small networks. An appropriate overestimation is necessary as the homeless vertices in an

ego network also need to be accommodated by some communities. Such overestimation is harmless as the false partial communities can be handled properly in the merger and post-processing steps that follow. It is reasonable to assume the number of partial communities is proportional to the ego network size. In the present work, the number of communities to be found is (over)estimated as $1/30$ of the network size, plus a lower bound of 5 and an upper bound of 20. The overestimation will not cause problem, as the true communities can be merged when they have a significant overlap. For the example in Fig. 1, we set out to find 10 communities and obtained 3 apparent partial communities (coloured purple, blue and green) and 2 possible ones (red and yellow). The green one actually consists of 3 highly overlapping communities which should be regarded as one. We merge a pair of partial communities if either one shares more than 30% the members of the other.

It is important to note that PCMA does *not* require a high accuracy in this step of finding all the partial communities. Any error generated in this step can be greatly reduced by the mergers and post-processing step, as discussed in Sec. II C.

[1] M. Girvan and M. E. J. Newman, Proc. Natl. Acad. Sci. U.S.A. **99**, 7821 (2002).
[2] S. Fortunato, Phys. Rep. **486**, 75 (2010).
[3] S. B. Seidman, Soc. Networks **5**, 97 (1983).
[4] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi, Proc. Natl. Acad. Sci. U.S.A. **101**, 2658 (2004).
[5] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, Nature **435**, 814 (2005).
[6] T. S. Evans and R. Lambiotte, Phys. Rev. E **80**, 016105 (2009).
[7] T. S. Evans and R. Lambiotte, Eur. Phys. J. B **77**, 265 (2010).
[8] Y.-Y. Ahn, J. P. Bagrow, and S. Lehmann, Nature **466**, 761 (2010).
[9] J. Baumes, M. Goldberg, M. Krishnamoorthy, M. Magdon-Ismail, and N. Preston, in *Proceedings of the IADIS International Conference on Applied Computing* (2005) pp. 97–104.
[10] A. Lancichinetti, S. Fortunato, and J. Kertész, New J. Phys. **11**, 033015 (2009).
[11] A. Lancichinetti, F. Radicchi, J. J. Ramasco, and S. Fortunato, PLoS ONE **6**, e18961 (2011).
[12] U. Raghavan, R. Albert, and S. Kumara, Phys. Rev. E **76**, 036106 (2007).
[13] S. Gregory, New J. Phys. **12**, 103018 (2010).
[14] J. Xie, B. K. Szymanski, and X. Liu, in *Proceedings of the 2011 IEEE 11th International Conference on Data Mining Workshops* (IEEE, 2011) pp. 344–349.
[15] J. Xie, S. Kelley, and B. K. Szymanski, ACM Comput. Surv. **45**, 1 (2013).
[16] G. Bianconi, P. Pin, and M. Marsili, Proc. Natl. Acad. Sci. U.S.A. **106**, 11433 (2009).
[17] A. Lancichinetti, F. Radicchi, and J. J. Ramasco, Phys. Rev. E **81**, 046110 (2010).
[18] B. Ball, B. Karrer, and M. E. J. Newman, Phys. Rev. E **84**, 036103 (2011).
[19] A. Condon and R. M. Karp, Random Struct. Alg. **18**, 116 (2001).
[20] L. Danon, A. Díaz-Guilera, J. Duch, and A. Arenas, J. Stat. Mech. , P09008 (2005).
[21] A. Lancichinetti, S. Fortunato, and F. Radicchi, Phys. Rev. E **78**, 046110 (2008).
[22] E. H. W. Xu, https://hwxu.net/research/pcma.
[23] S. Gregory, J. Stat. Mech. , P02017 (2011).